Tecnológico de Monterrey

Métodos Numéricos

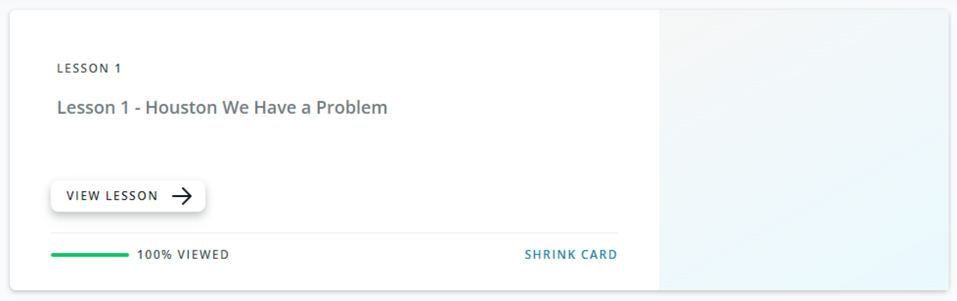
Mtro. Adolfo Centeno T

Differential Equations in Action - Lesson 1

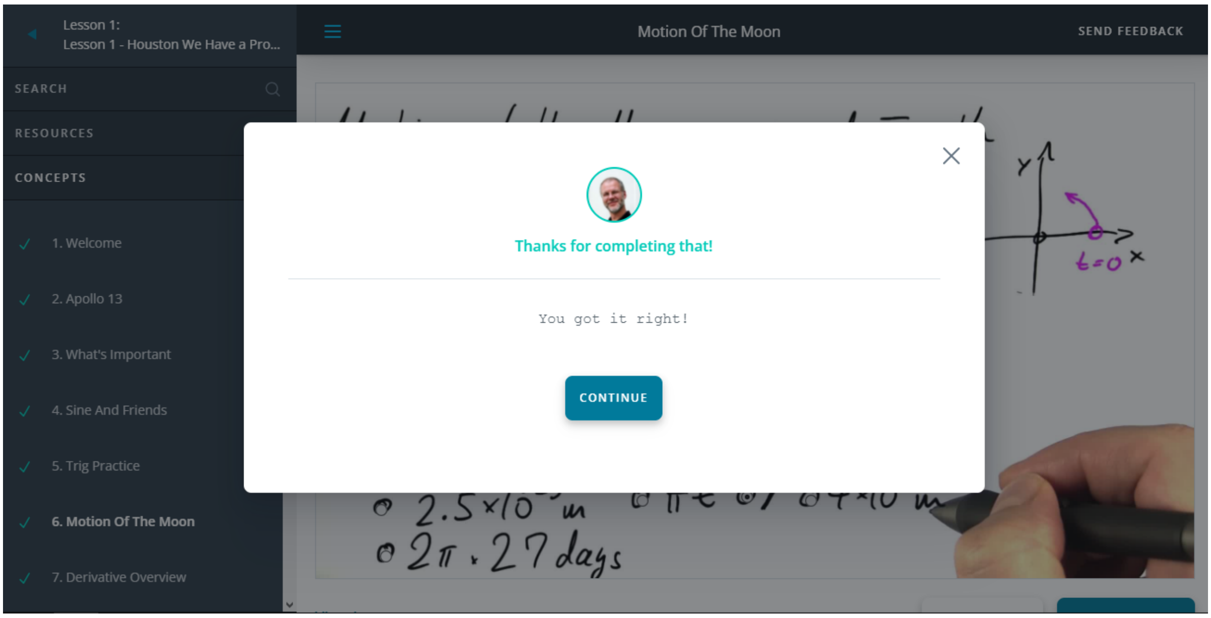
Integrantes:

Maximiliano Macotela Nava, Luz Karen Hernández Hernández, Andres Serrato Barrera, Julio Rodríguez Salcedo, César David Rosales Álvares, Jim Kevin Holguín Rodríguez

Lesson 1 - Houston, We Have a Problem



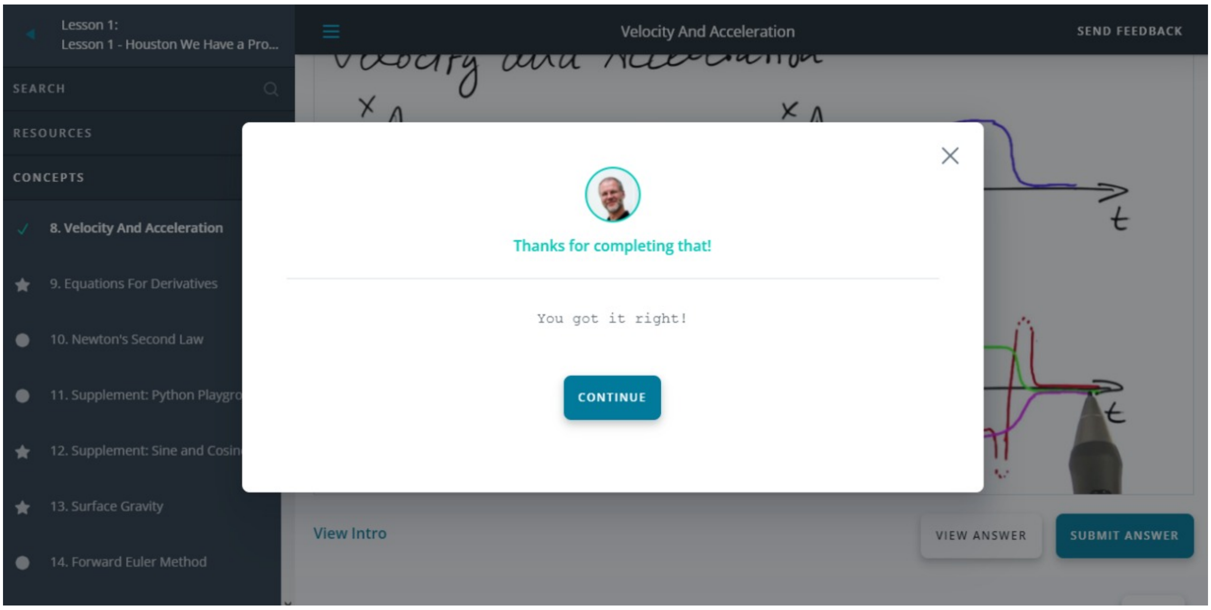
6. Motion Of The Moon



Thanks for completing that!

You got it right!

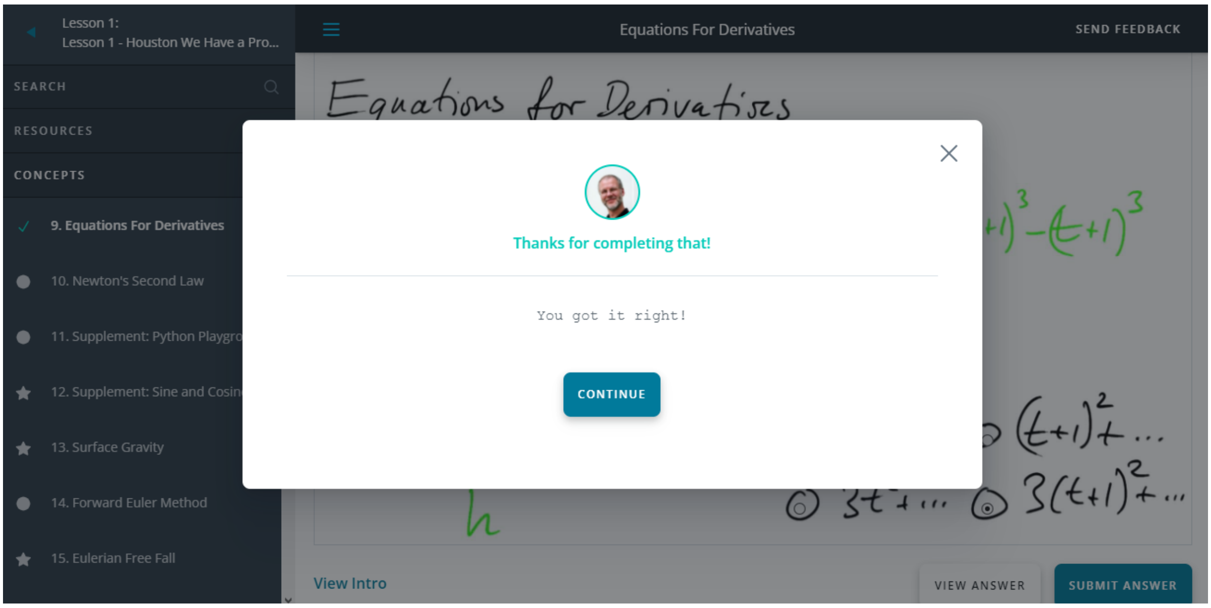
8. Velocity And Acceleration


Lesson 1:
Lesson 1 - Houston We Have a Pro...
Velocity And Acceleration
SEND FEEDBACK
SEARCH
RESOURCES
CONCEPTS
✓ 8. Velocity And Acceleration
★ 9. Equations For Derivatives
● 10. Newton's Second Law
● 11. Supplement: Python Playgro
★ 12. Supplement: Sine and Cosin
★ 13. Surface Gravity
● 14. Forward Euler Method
x
x
t
t
CONTINUE
View Intro
VIEW ANSWER
SUBMIT ANSWER

Thanks for completing that!

You got it right!

9. Equations For Derivatives



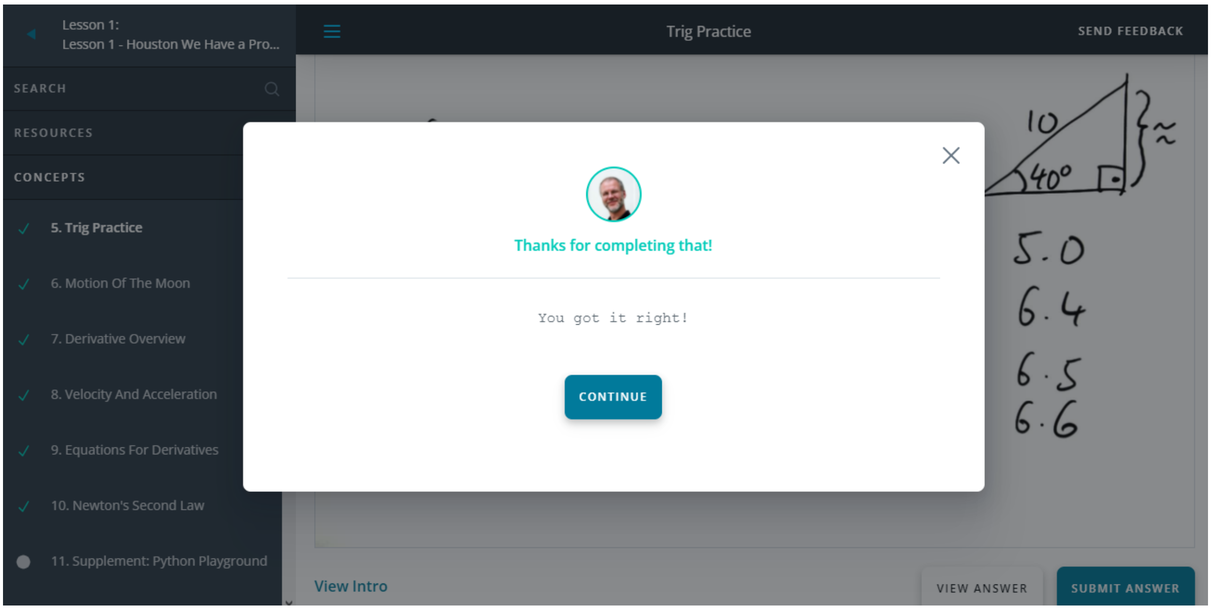
Lesson 1:
Lesson 1 - Houston We Have a Pro...

Equations For Derivatives
SEND FEEDBACK
SEARCH
RESOURCES
CONCEPTS
✓ 9. Equations For Derivatives
● 10. Newton's Second Law
● 11. Supplement: Python Playgro
★ 12. Supplement: Sine and Cosin
★ 13. Surface Gravity
● 14. Forward Euler Method
★ 15. Eulerian Free Fall
Equations for Derivatives
$+1)^3-(t+1)^3$
$(t+1)^2 t \ldots$
$3t^2 t \ldots$
$3(t+1)^2 t \ldots$
h
View Intro
VIEW ANSWER
SUBMIT ANSWER

Thanks for completing that!

```
You got it right!
```

10. Newton's Second Law



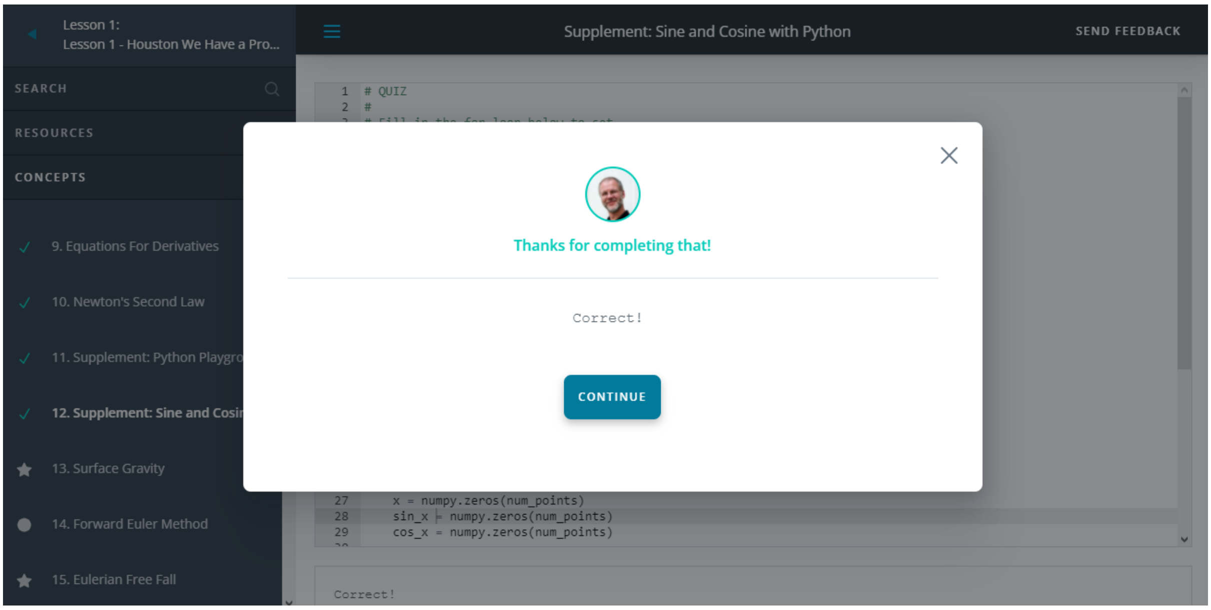
Lesson 1:
Lesson 1 - Houston We Have a Pro...

Thanks for completing that!

```
You got it right!
```

12. Supplement: Sine and Cosine with Python

```
import math
from udacityplots import *


def sin_cos():
    num_points = 50

    x = numpy.zeros(num_points)
    sin_x = numpy.zeros(num_points)
    cos_x = numpy.zeros(num_points)

    for i in range(num_points):
        x[i] = 2. * math.pi * i / (num_points - 1.)
        sin_x[i] = math.sin(x[i])
        cos_x[i] = math.cos(x[i])
    return x, sin_x, cos_x

x, sin_x, cos_x = sin_cos()

@show_plot
def plot_me():
    matplotlib.pyplot.plot(x, sin_x)
    matplotlib.pyplot.plot(x, cos_x)
plot_me()
```

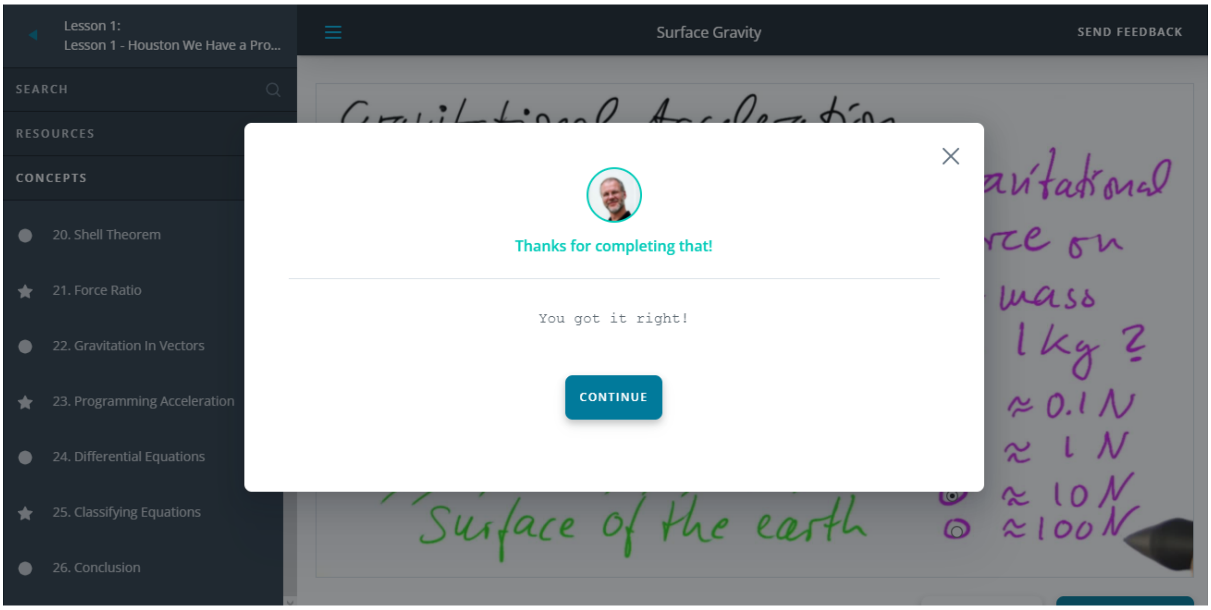

Lesson 1:
Lesson 1 - Houston We Have a Pro...
SEARCH
RESOURCES
CONCEPTS
✓ 9. Equations For Derivatives
✓ 10. Newton's Second Law
✓ 11. Supplement: Python Playgro
✓ 12. Supplement: Sine and Cosin
13. Surface Gravity
14. Forward Euler Method
15. Eulerian Free Fall
Supplement: Sine and Cosine with Python
SEND FEEDBACK
Thanks for completing that!
CONTINUE

```
# QUIZ
#
    x = numpy.zeros(num_points)
    sin_x = numpy.zeros(num_points)
    cos_x = numpy.zeros(num_points)
```

Correct!

Correct!

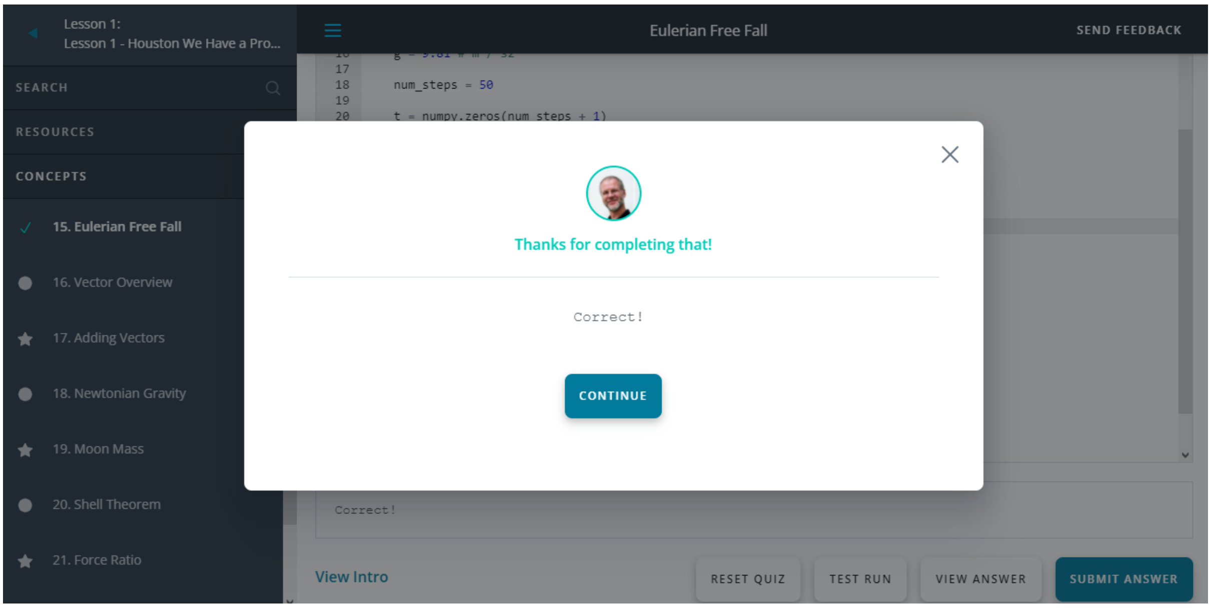
13. Surface Gravity



You got it right!

15. Eulerian Free Fall

```
from udacityplots import *

def forward_euler():
    h = 0.1 # s
    g = 9.81 # m / s2

    num_steps = 50

    t = numpy.zeros(num_steps + 1)
    x = numpy.zeros(num_steps + 1)
    v = numpy.zeros(num_steps + 1)

    for step in range(num_steps):
        t[step + 1] = h * (step + 1)
        x[step + 1] = x[step] + h * v[step]
        v[step + 1] = v[step] - h * g
    return t, x, v

t, x, v = forward_euler()

@show_plot # Remove this line when running locally
def plot_me():
    axes_height = matplotlib.pyplot.subplot(211)
    matplotlib.pyplot.plot(t, x)
    axes_velocity = matplotlib.pyplot.subplot(212)
    matplotlib.pyplot.plot(t, v)
    axes_height.set_ylabel('Height in m')
    axes_velocity.set_ylabel('Velocity in m/s')
    axes_velocity.set_xlabel('Time in s')

plot_me()
```



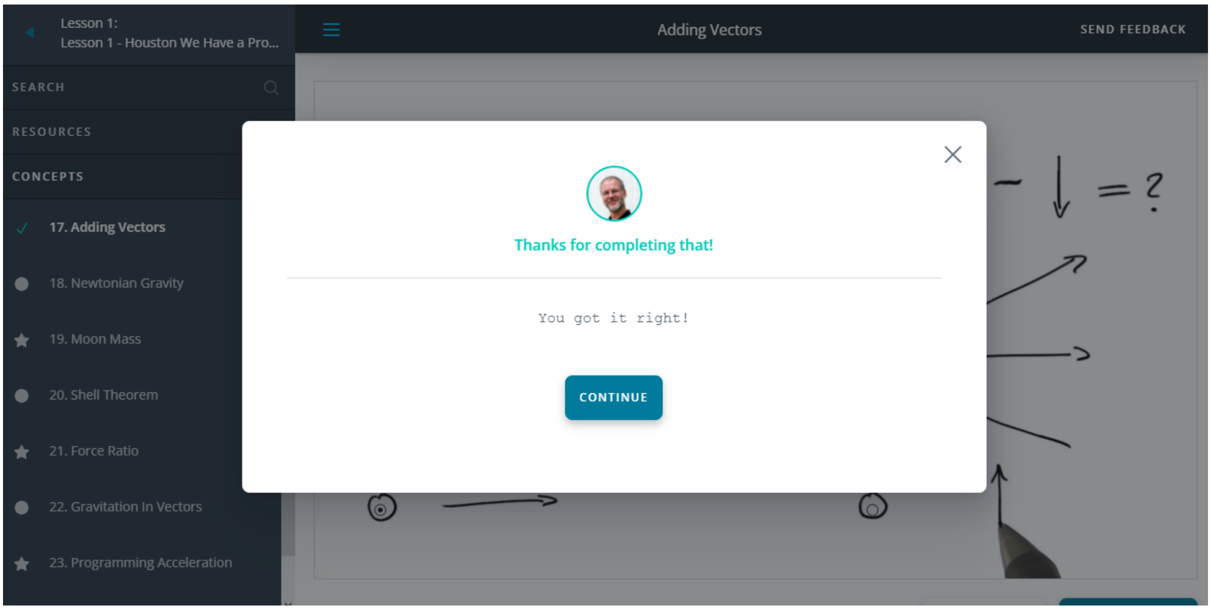
```
    g = 9.81 # m / s2

    num_steps = 50

    t = numpy.zeros(num_steps + 1)
```

Thanks for completing that!

```
Correct!
```

```
Correct!
```

17. Adding Vectors

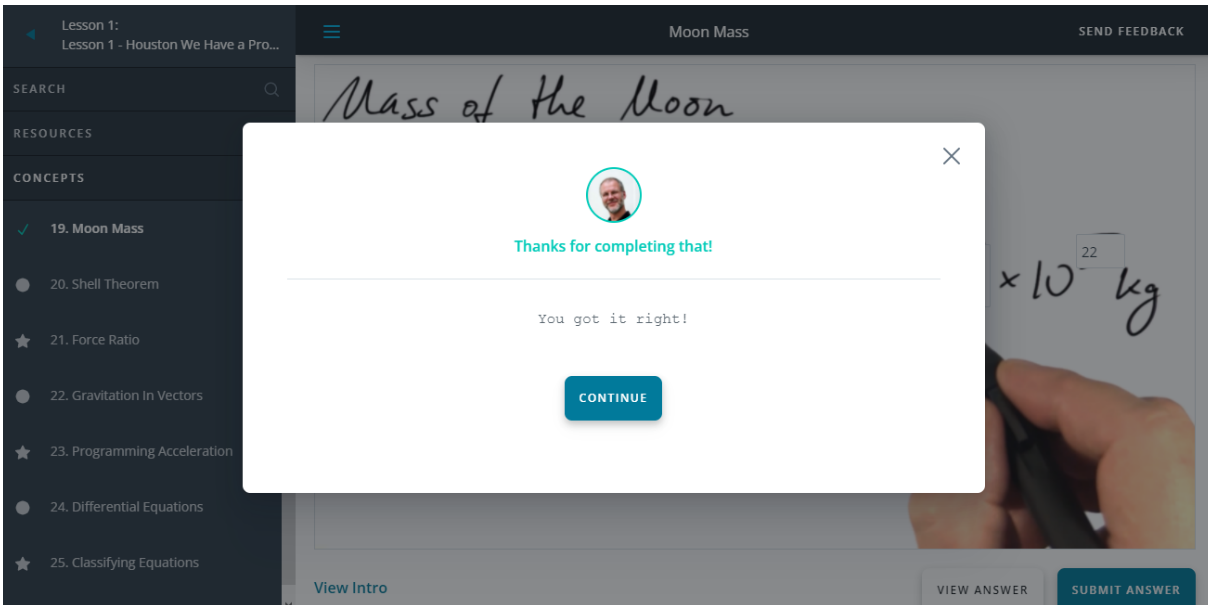


Lesson 1:
Lesson 1 - Houston We Have a Pro...

Thanks for completing that!

You got it right!

19. Moon Mass

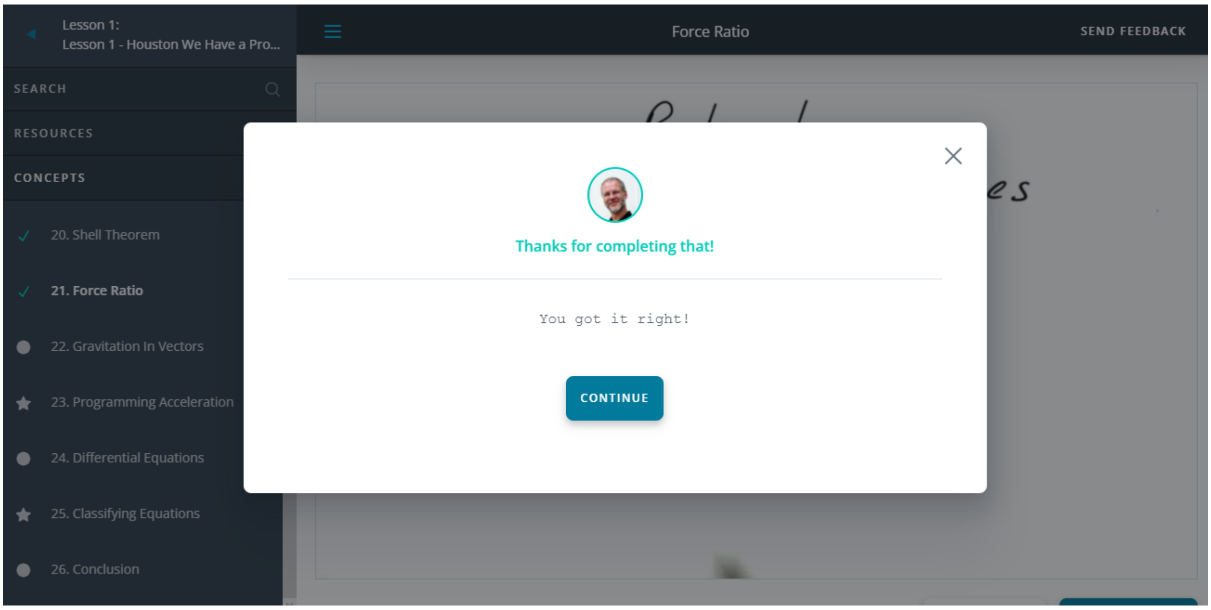


Lesson 1:
Lesson 1 - Houston We Have a Pro...

Thanks for completing that!

You got it right!

21. Force Ratio

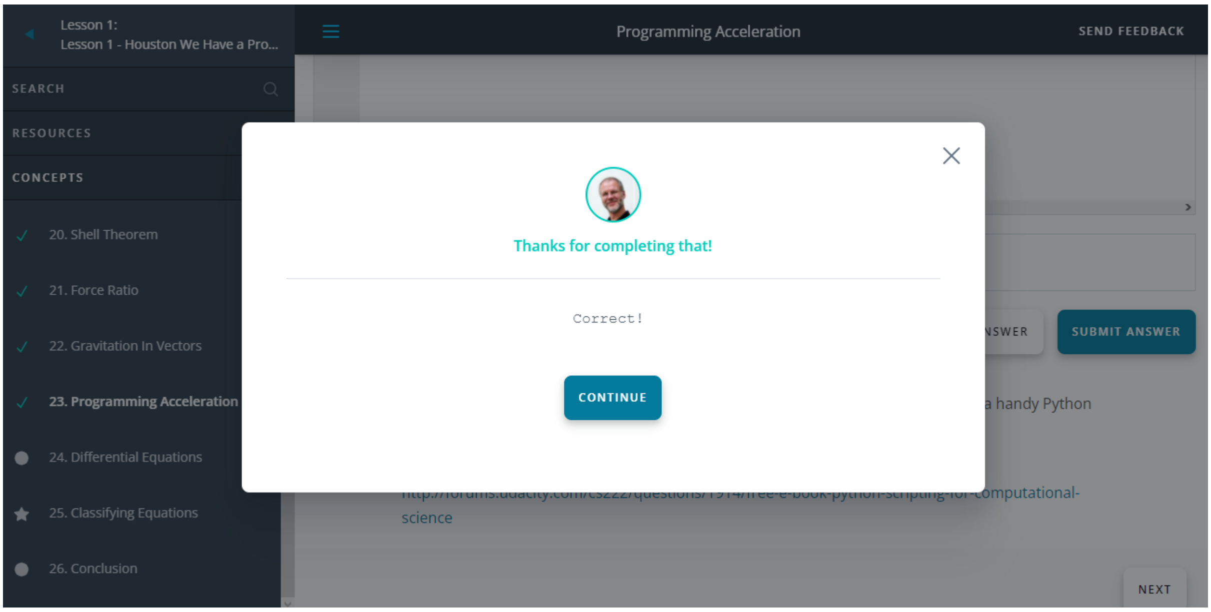

Lesson 1:
Lesson 1 - Houston We Have a Pro...
Force Ratio
SEND FEEDBACK
SEARCH
RESOURCES
CONCEPTS
✓ 20. Shell Theorem
✓ 21. Force Ratio
● 22. Gravitation In Vectors
★ 23. Programming Acceleration
● 24. Differential Equations
★ 25. Classifying Equations
● 26. Conclusion
CONTINUE

Thanks for completing that!

You got it right!

23. Programming Acceleration

```
import numpy

earth_mass = 5.97e24 # kg
moon_mass = 7.35e22 # kg
gravitational_constant = 6.67e-11 # N m2 / kg2

def acceleration(moon_position, spaceship_position):
    vector_to_moon = moon_position - spaceship_position
    vector_to_earth = - spaceship_position
    return gravitational_constant* (earth_mass /
numpy.linalg.norm(vector_to_earth)**3 * vector_to_earth + moon_mass /
numpy.linalg.norm(vector_to_moon)**3 * vector_to_moon)
```



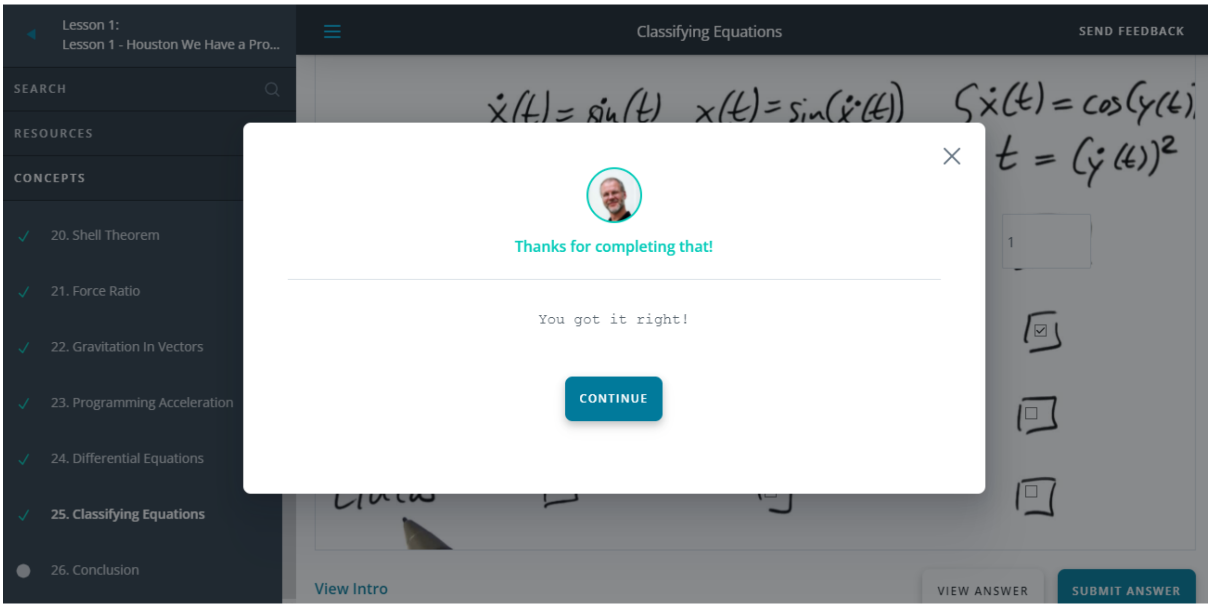
Thanks for completing that!

Correct!

a handy Python

http://forums.udacity.com/cs222/questions/1914/free-e-book-python-scripting-for-computational-science

25. Classifying Equations



Thanks for completing that!

You got it right!